





Graphic Era
deemed to be **University**
DEHRADUN

PROJECT AND TEAM INFORMATION

Project Title

AN EFFECTIVE APPROACH TO NETWORK PENETRATATION TESTING

Student/Team Information

Team Name: Team #	The Cyber Squad CYBER-IV-T007
Team member 1	Dwivedi,Shubh-230112175 230112175@geu.ac.in 
Team member 2	Gupta,Riya-23022643 23022643@geu.ac.in 

Team member 3	<p>Kunwri,Jahnavi-23022824</p> <p>23022824@geu.ac.in</p>  A portrait of a young woman with long dark hair, wearing a light blue school shirt and a red and white striped tie. She is standing in front of an orange patterned curtain.
Team member 4	<p>Vats,Disha-23022703</p> <p>23022703@geu.ac.in</p>  A portrait of a young woman with dark hair and glasses, wearing a light blue school shirt and a red and white striped tie. She is standing in front of a plain white background.

PROJECT PROGRESS DESCRIPTION

Project Abstract

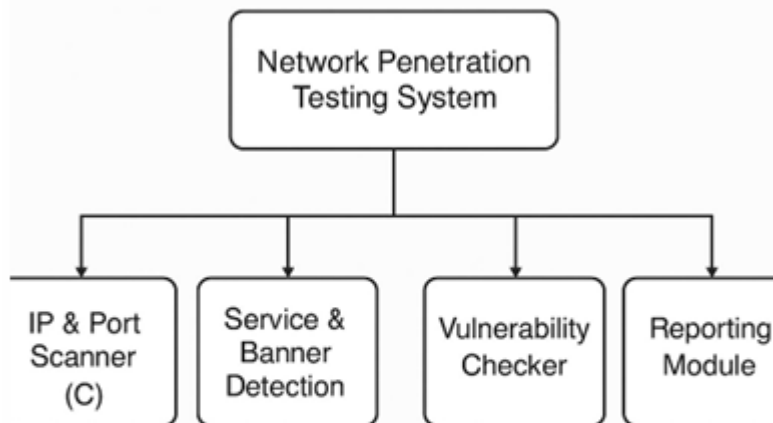
NetGuard is a lightweight, Python-based network penetration testing tool designed to simplify and automate core security assessment tasks. Built with a user-friendly GUI using tkinter, the tool allows users to scan IP ranges for open ports, grab service banners, detect known vulnerabilities, and analyze password strength—all within a single interface. Its multithreaded port scanning mechanism ensures fast performance over subnets, while the banner grabbing and CVE-based detection modules help identify potentially risky services. Each module is integrated seamlessly into the interface, enabling real-time updates and comprehensive output without overwhelming the user.

The tool also features a manual password strength checker and a report generator that consolidates all results into a timestamped text file. With a modular architecture and reliance on standard Python libraries such as socket, ipaddress, and threading, NetGuard remains cross-platform and easy to maintain. Designed for educational use, cybersecurity students, and small organizations, this project offers an effective starting point for network reconnaissance and vulnerability evaluation without requiring advanced technical knowledge or expensive commercial tools.

Updated Project Approach and Architecture

The NetGuard system adopts a modular approach, with each functionality—such as scanning, banner grabbing, vulnerability detection, and reporting—encapsulated in independent Python functions. The application uses multithreading to efficiently scan ports 1 to 99 across an IP subnet, significantly improving speed and responsiveness. Banner grabbing is implemented over TCP sockets to retrieve service information, which is then matched against a dictionary of known vulnerable software for quick threat detection. A manual password strength checker is also included to evaluate weak or common passwords based on predefined rules.

The tool is built entirely using Python's standard libraries and features a user-friendly GUI designed with tkinter. This interface provides a menu-driven experience, allowing users to perform actions like subnet scanning, banner inspection, vulnerability checks, and report generation with a single click. The reporting module consolidates findings into a readable .txt file, summarizing port scan results, service banners, detected vulnerabilities, and password analysis. The architecture is scalable, cross-platform, and designed for future enhancements such as CSV export, extended CVE databases, and thread pool optimization for larger subnets.



The diagram illustrates a **Network Penetration Testing System** divided into four key functional modules. The **IP & Port Scanner**, developed in C, identifies active hosts and open ports in a network. The **Service & Banner Detection** module gathers details about the services running on these ports and their versions. The **Vulnerability Checker** evaluates identified services for known weaknesses and exploits. Finally, the **Reporting Module** compiles and presents the findings in a structured format, aiding in the assessment and remediation of network security risks.

Tasks Completed

Task Completed	Team Member
Implemented Password Strength Checker	Shubh Dwivedi
Implemented C-based Port Scanner using subprocess	Shubh Dwivedi
Developed Python-based Port Scanning Module	Riya Gupta
Designed Menu Driven Interface	Riya Gupta
Implemented Banner Grabbing Module	Jahnvi Kunwri
Developed Vulnerability Detection Logic	Disha Vats
Created Report Generation Module	Disha Vats

Challenges/Roadblocks

One major challenge we faced was **handling socket timeouts** during port scanning. Some IPs in the subnet didn't respond, which caused unnecessary delays. We resolved this by reducing the timeout value for socket connections, allowing the scanner to skip unresponsive ports more quickly.

Banner grabbing presented difficulties as many services either did not return banners or sent incomplete data. This affected vulnerability detection accuracy. To handle this, we added error handling and are currently developing a fallback method using port-to-service mapping.

Performance issues arose when scanning large subnets. The tool created too many threads, overwhelming system resources. We limited the number of scanned ports for now and plan to introduce a thread pool system to better manage concurrency.

Initially, the **GUI would freeze** during scans due to blocking operations. This affected usability. We fixed it by using Python's threading module to run scans in the background, keeping the interface responsive while showing live updates.

The **password checker** had to be manual due to security and input concerns. Designing it to be both simple and effective required a clear approach. We used a list of weak passwords and added rules for minimum length to identify poor passwords easily.

Integrating the optional **C-based port scanner** posed cross-platform challenges. On some systems, it failed to execute due to path or permission issues. We focused on the Python-based version for full compatibility within the GUI.

Lastly, **report formatting** was tricky with multithreaded outputs arriving in random order. We implemented synchronized data collection and designed a clean report structure that includes port scans, banners, vulnerabilities, and password analysis in separate sections.

Future Scope

The NetGuard penetration testing tool has strong potential for future development and real-world application. One key enhancement is the expansion of the internal vulnerability database by integrating it with publicly available CVE repositories or APIs like NIST NVD. This would allow for real-time vulnerability matching and broader detection capabilities. Additionally, incorporating machine learning algorithms to classify banners and predict unknown vulnerabilities could significantly improve the tool's accuracy and intelligence in detecting emerging threats.

Another major improvement involves optimizing performance for large-scale networks. Implementing thread pools, asynchronous I/O, or even parallel processing using multiprocessing could make the tool suitable for enterprise-level scanning. A planned feature includes exporting reports in CSV or PDF formats for better integration into professional workflows. Lastly, integrating authentication bypass checks, brute-force attack simulation, and advanced service fingerprinting would further evolve NetGuard into a more complete penetration testing suite, expanding its use beyond educational environments to more advanced security assessments.

Project Outcome/Deliverables

The NetGuard project successfully delivered a functional and user-friendly Python-based penetration testing tool that integrates multiple security assessment features into a single GUI. The tool performs multithreaded port scanning, banner grabbing, basic vulnerability detection based on known software signatures, and manual password strength analysis. It allows users to enter a subnet or IP address, conduct tests efficiently, and view real-time results through a responsive tkinter interface. The final product is suitable for students, educators, and small organizations seeking to understand basic network security concepts.

As part of the deliverables, the project includes the complete source code, a documented usage guide, sample penetration test reports (.txt), and test logs. The tool's modular structure ensures easy future updates, while the generated reports provide clear summaries of scan results, detected banners, potential vulnerabilities, and password evaluations. Optional features such as CSV report export and a larger vulnerability database are under development and may be added in future releases, further enhancing the practicality and educational value of NetGuard.

Progress Overview

The NetGuard project has been fully completed, with all planned features successfully implemented, tested, and integrated into a user-friendly GUI. The tool now supports complete functionality including multithreaded port scanning, service banner grabbing, known vulnerability detection, manual password strength analysis, and comprehensive report generation. All modules function cohesively within the tkinter-based graphical interface, offering real-time feedback and seamless user interaction.

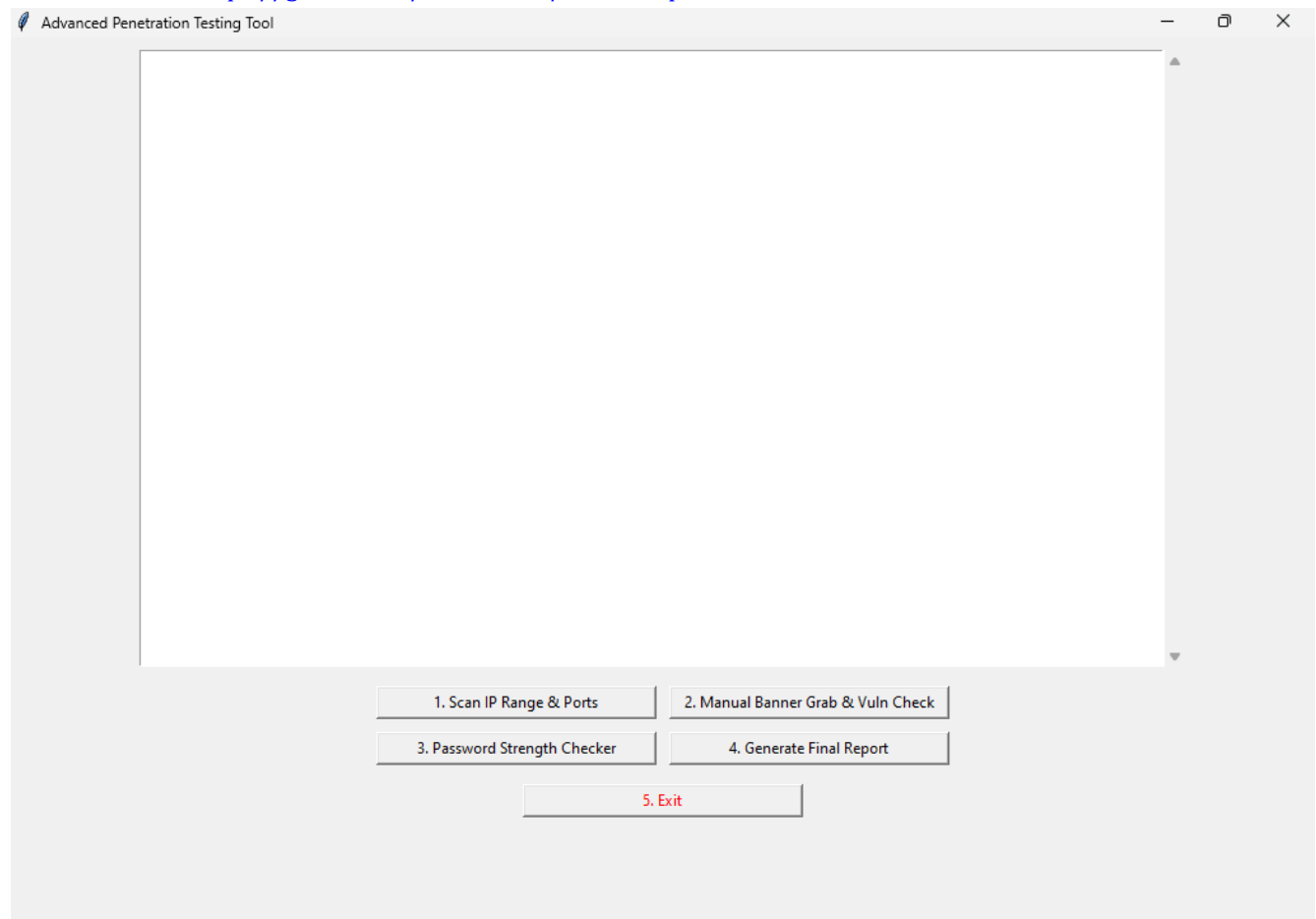
In addition to the core features, optional components such as structured report formatting, GUI responsiveness improvements, and error handling have also been implemented. The codebase is clean, modular, and well-documented, allowing for future enhancements and easy maintenance. With successful internal validation and testing on various IP ranges and ports, the tool is ready for demonstration and deployment. The team has met all deliverables as outlined, and the project stands as a robust, educational, and practical solution for basic network penetration testing.

Codebase Information

The complete source code for the NetGuard project is maintained in a Git-based version control system to ensure collaborative development and traceability. The repository follows a modular structure, with separate files and functions for each major feature—port scanning, banner grabbing, vulnerability detection, password checking, report generation, and the GUI interface. Each module was committed by the respective team member responsible for its development, ensuring clean code ownership and accountability.

Key commits in the repository highlight milestones such as the implementation of multithreaded scanning, integration of the vulnerability database, GUI setup using tkinter, and final testing scripts. The repository also includes a README file containing setup instructions, usage guidelines, and sample outputs for testing. The main working branch is main, and all testing and feature additions were merged after verification. The project is publicly accessible for review and further development.

Git Hub Link: https://github.com/shubh9125/network_penetration_tester



Testing and Validation Status

Test Type	Status (Pass/Fail)	Notes
Port scan on localhost	Pass	Detected expected open ports on test machine
Banner grabbing from open ports	Pass	Some services returned no banners : fallback logic is being implemented
Vulnerability detection matching	Pass	Successfully flagged known banners like vsFTPD 2.3.4 and Apache/2.2.8
Password strength checker	Pass	Correctly identified weak and short passwords
C-based port scanner execution	Pass	Successfully executed via subprocess: output verified
Menu navigation and CLI testing	Pass	All menu option work correctly and call respective modules
Report Generation	Pass	Report file (pentest_report.txt) generated correctly with all scan and check data.

Deliverables Progress

The advanced penetration testing tool is complete, featuring multi-threaded port scanning, banner grabbing, vulnerability detection, and a password strength checker—all integrated into a responsive Tkinter GUI. The tool provides real-time scan results and generates a detailed report summarizing findings.

Testing confirms accurate detection of open ports, banners, vulnerabilities, and weak passwords. The GUI handles user input smoothly and remains stable during scans. Error handling is in place, and the project meets all key requirements, with room for future enhancements.

